Contribution to the DOE National Collaboratories Program Investigator's Meeting, August 10-12 at ANL.

Grid Troubleshooting

Brian Tierney, Dan Gunter, Chuck McParland, Doug Olson
Lawrence Berkeley National Laboratory

A major barrier to widespread acceptance of Grid technology is the difficulty of debugging and troubleshooting Grid applications. Grid applications have heterogeneous widely distributed components, each with their own style of logging and error reporting. Error messages are scattered across a large number of systems, and log file formats are all different. Collecting and combining all this data is a very difficult and tedious, if not impossible, process [6].

Performance tuning a Grid application is even more difficult than debugging because it requires more varied and detailed information. Often a Grid application is working fine, but is running slower than expected. Is the problem related to network load, disk load, system load, or poorly designed software?

Addressing this problem for today's and tomorrow's grid applications involves research of some new concepts, like Grid ID's, as well as research that can only be carried out in the context of a large-scale production Grid, such as how to collect the "stack trace" for a Grid ID using the available resource and information discovery services. Integrating these required new concepts into the middleware deployment such as VDT provides the ability to carry out this research on the Physics community's Grid fabric.

Both instrumentation data (information about the progress of a piece of software at a particular time) and monitoring data (information about a hardware component such as CPU, disk, or network at a particular time) are needed for troubleshooting [1]. However a great deal of infrastructure is needed to for the instrumentation and monitoring data to be accurate, available, portable, and useful. Some support for this exists, such as NetLogger [2] and the Globus MDS [1].

There are a number of missing pieces that are needed to make this work. These include:

- Grid ID's: all instrumentation and monitoring log events for a given Grid Job must contain a Grid Job ID
- A common data model: all instrumentation and monitoring data should share a common data model, so that it can be added to databases, indexed, and queried against.
- Automatic instrumentation: Tools to automatically add instrumentation to applications and middleware are needed, along with tools to publish this data to the Grid.
- Resource discovery: there must be a way to easily locate all the instrumentation and monitoring data related to given Grid Job.
- Analysis Tools: The amount of instrumentation and monitoring data will be overwhelming without good analysis tools.

These are all discussed in more detail below.

Grid IDs

As described in [3], adding a Grid ID to all instrumentation events is a simple mechanism to provide the ability to correlate data from a variety of sources related to a single Grid job. Of course some monitoring data, such as CPU load, is not tied to a particular Grid Job, and will therefore not have a Grid ID. This type of data will have to be correlated using timestamps and/or using out-of-band information.

Common Data Model

A common log format such as that provide by NetLogger is very useful, but a common data model is a fundamental requirement for analysis of independent data sources. Mapping between disparate data models is difficult, even more difficult than format translations. Within the realm of monitoring data, we believe that we can arrive at a common data model that combines adequate expressiveness with efficient and natural representations in at least the two most important contexts: in relational databases and as serialized bytes "on the wire".

Progress in this area has been made from two directions: standards within the Global Grid Forum (GGF), and implementations within the NetLogger Toolkit. In the GGF, we participated in the Discovery and Monitoring Event Descriptions (DAMED) working group, which produced a description of how the

attributes of monitoring data could be represented in a consistent and straightforward way [4]. In the NetLogger Toolkit we have provided an efficient wire representation that maps to the DAMED data model.

Automatic Instrumentation Tools

Based on our experience with supporting the NetLogger Toolkit for the past several years, busy programmers rarely get around to instrumenting their code. To overcome this, we need automatic instrumentation tools that can be applied to deployed software components (not just used in the development environment). This way, non-"owners" of the software can easily add instrumentation. For compiled languages like C and Fortran, these tools should work on object files and not require access to source code. For interpreted languages like Python and Java, the introspection capabilities of the language could be used to provide run-time control of instrumentation points. Even when the source code is available, tools of this type would dramatically speed up the iterative process of instrumenting applications and middleware.

Although much progress has been made in providing better user interfaces for submitting tasks to Grid systems, these computational environments are typically remote and difficult to interact with directly. As a result, many of the techniques used for local application debugging actual are unavailable to users. While it is usually possible to print performance logs to a temporary file or the user's console, this technique is often too cumbersome for users attempting to monitor behavior of data analysis or simulation tasks. A set of standardized objects or libraries could allow users to "decorate" their code with values to be monitored and performance metrics to be graphed. If transport of this monitoring data were transparent to the user, it could allow users to verify and track program execution from any suitable location (desktop, lab, home, etc.). In addition, if this data were logged to a facility-maintained database, users, with little additional effort, would be able to perform historical comparisons of both program and CPU performance. In short, much of the "remoteness" of the Grid execution environment would disappear.

Data Discovery

In some cases instrumentation data will be sent directly to the user, and in other cases instrumentation and monitoring data will be sent to one of a number of monitoring data repositories that will be part of the Grid. There needs to be an easy way to collect all this data to a single location for analysis. We expect that the frequency of instrumentation and monitoring data updates will be much greater than the frequency of discovery queries and users are interested in timely information. Because of this, we envision using P2P-based resource discovery tools such as LBNL's P2PIO for on-demand discovery of this data.

Analysis Tools

The point of gathering monitoring data is not to have it, but draw conclusions from it. There must be easy-to-use but flexible tools to make sense of all the data. This data must be available and in a form that is useful for not just developers, but also system administrators and end-users.

Conclusion

Troubleshooting and performance analysis of applications running in a Grid environment is still very difficult. However we believe that this problem can be solved by a combination of the above ideas.

References

1. K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman. *Grid Information Services for Distributed Resource Sharing*. Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press, August 2001.
2. D. Gunter, B. Tierney, C. E. Tull, V. Virmani, *On-Demand Grid Application Tuning and Debugging with the NetLogger Activation Service*, 4th International Workshop on Grid Computing (Grid2003), November 2003.
3. D. Gunter, Keith Jackson, David Konerding, Jason Lee, Brian Tierney, *Essential Grid Workflow Monitoring Elements*, submitted to Grid 2004.
4. D. Gunter and J. Magowan, *An analysis of "Top N" Event Descriptions*, Global Grid Forum Informational Document: GGD-I-025, 2004.
5. B. Tierney and J. Hollingsworth, "Instrumentation and Monitoring", Chapter 20, The Grid Blueprint for a New Computing Infrastructure, 2nd Edition, Elsevier, 2003.
6. Troubleshooting and Fault Tolerance in Grid Environments Workshop, Chicago, Dec 11th 2002, PPDG-26, http://www.ppdg.net/docs/Papers/Troubleshooting.pdf